POSTFLOW

POSTFLOW is a tool for extracting data from a DPLR restart file and formatting it appropriately for presentation or further post-processing. POSTFLOW is a data-extractor *only*, it does not perform any data visualization. However, POSTFLOW does include some rather powerful options that simplify the data-extraction process and ensure that the user can quickly obtain exactly the data they need in the format they want from a given simulation.

When the restart file is generated by the CFD code, all CFD input deck flags and physical modeling parameters that were used in the simulation are written to the restart file. These flags are not read by DPLR2D or DPLR3D on input, but they are read by POSTFLOW when the restart file is processed. In this manner, the user is ensured that the data are post-processed in a manner consistent with the way in which the data were generated. The original CFD input and physical modeling database files are never read by POSTFLOW, and need not be present in order to do post-processing. Therefore, even if the CFD input deck or any of the physical properties databases are later changed or lost, the user can always extract physically consistent data from the restart file.

Although the format of the restart file changes occasionally (possible format changes are signaled by a change in the minor release number of the code), backward compatibility is always maintained. The version of DPLR that was used to run the simulation is also written to the restart file, and POSTFLOW will detect this version when post-processing. If POSTFLOW is used to process a restart file from an older version of the package, an informational message will be output to the screen, but otherwise POSTFLOW will function normally.

Running POSTFLOW

POSTFLOW is run from the command line. The user first prepares the input deck, either by starting from a similar case or by following the rules discussed in the following section. In order to execute the run, type:

```
postflow < post.inp
```

at the command line, where "`post.inp`" is the name of the input deck. When POSTFLOW is executed diagnostic output will be echoed to the screen in order to provide feedback on the action(s) being performed. Any warning messages will also be echoed to the screen. If a fatal error is detected during execution, a descriptive message will be echoed to the screen and execution will terminate. POSTFLOW always runs in serial on a single processor, regardless of the number of processors that were used to run the simulation. However, it can easily be compiled to run within MPI if required for the destination machine.

The sample input deck presented in the following section replicates that for one of the sample problems ("`Neptune`") provided with this distribution. When POSTFLOW is executed on this sample problem, the output is:

```
*******************************************
 postflow
 NASA Ames Version 3.05.0
 Mike Wright     last modified: 04/07/06
 *******************************************

 restart file format: NASA Ames Version 3.05.0
 solution run at: Fri Mar 31 08:31:34 2006

 run in   700 iterations in 5.00E+03 seconds

 CPP-macro settings enabled during run:
       PARKTEXP=0.50

 input  ns =  3;  nev =  1
 number of blocks =  2
 file dimension   =  3

 extracting the following BCs :    19
 note that extraction of pointwise BCs not supported yet

 output variables=x,y,z,p,T,M,res

 running in high memory mode

 interpolating grid to cell centers

 processing grid variable  1  2  3
 processing flow variable  1  2  3  4  5  6  7  8  9

 block # 1: nx =   32; ny =   16; nz =   64
      zone t=BC19          i=  34 j=   1 k=  66

 processing grid variable  1  2  3
 processing flow variable  1  2  3  4  5  6  7  8  9

 block # 2: nx =   48; ny =   64; nz =   64
      zone t=BC19          i=  50 j=   1 k=  66

      zone t=BC19          i=  50 j=   1 k=  66


 writing tecplot file: neptune.plt

 using grid file: neptune.pgrx
 using flow file: neptune.pslx
```

As can be seen, the screen output provides a step-by step discussion of the actions performed during execution, and can be used to ensure that the actions performed match those desired.

Sample Input Deck

A sample input deck for POSTFLOW is shown below. A brief description of each of the flags, along with allowable settings, is provided in the following section. Detailed discussions of some of the more complex options follow. Additional examples of POSTFLOW input decks can be found in the sample problems that are distributed with the DPLR package; it is recommended that the user run through each of the provided examples after reading this chapter and examine the output of POSTFLOW for each case.

```
Input file for postflow

imemmode itruev
  2        1

inrest   ingrid  inbcf   ouform  iwrtd
 11        0       0       6       0

interp   nzones   isep   istyp
  1       10       0        1

lref   aref   xmc   ymc   zmc   imrx  imry  imrz
 1.0    1.0   0.0   0.0   0.0    0     1     0

iwind   cxs   cys   czs
  0     1.0   0.0   0.0

iexbc       <== list of BC numbers to extract from dataset
 19,26

ivarp       <== list of variable numbers to extract from dataset
 0 110 120 154 999

Tecplot/plot3d zone information:
iwrt ifac imin imax jmin jmax kmin kmax  bkmin  bkmax  zonetitle
0,   0,   1,  -1,   1,  -1,   1,  -1,    1,    -1    'flow2d'
-1,  0,   1,  -1,   1,  -1,   1,  -1,    1,    -1 'terminator'

fname,pname,(gname),(bname)
'neptune'
'neptune'
```

Summary of Input Flags

Input flags will be described in the order in which they appear in the input deck. A full description of some of the more complex option will be deferred until later sections.

Some of the flags presented below are present for future expansion of the capabilities of POSTFLOW and are not currently used. These will be indicated as they appear.

`imemmode`

POSTFLOW has been written to run efficiently for large problems. One of the ways in which this efficiency is achieved is by reading the entire set of stored variables into memory for a given physical grid block. This data is then processed and a subset is extracted based on the user's preferences. Since POSTFLOW is a serial code, an obvious consequence of this is that the machine on which the post-processing is done must have sufficient memory to hold all flow variables for the largest physical block in the simulation. There are certain cases when the available memory is not sufficient. In this case, a low memory mode is available, in which the flow variables are read one at a time, processed individually, and then purged from memory before continuing to the next variable. This requires much less memory, but it can take significantly longer. In addition, when this low memory mode is used, certain features, such as exact viscous fluxes (see below), are not available. POSTFLOW will issue a warning message when low memory mode is selected and a requested feature is not compatible. It is recommended that high memory mode be used whenever possible. Possible values for `imemmode` are:

1	low memory mode
2	high memory mode (recommended)

`itruev`

Derivative-based quantities (such as skin friction or heat transfer) can be computed either using an accurate second-order central differenced discretization of the Navier-Stokes flux terms (equivalent to the method used in DPLR, see Section XX of the Reference Manual for more information) or with a "quick-and-dirty" first-order approximation. For example, a first-order approximation for the convective component of the heat flux at a non-catalytic surface is given by:

$$q = \kappa \nabla T = \kappa \frac{\Delta T}{\Delta \eta} \tag{1}$$

where η is the distance from the first cell center away from the wall to the face center on the wall itself. The simple expression above assumes that the body-normal grid lines are truly orthogonal to the surface; a slight generalization must be applied to account for any non-orthogonality:

$$q = \kappa \frac{\Delta T}{\Delta \eta} cos(90 - \beta) \tag{2}$$

where β is the local angle (in degrees) between the grid lines normal and parallel to the body surface. Note that Eq. (2) reduces to Eq. (1) if the grid lines are truly orthogonal (β = 90°). Similar expressions can be constructed for all other derivative based quantities. For most applications the difference between the accurate and approximate calculation of these derivative based quantities is very small (on the order of 1% or less), and either method is acceptable. However, it is preferred to use the exact representation of these values (`itruev` = 1) whenever possible for consistency with the CFD code. It should be noted that when post-processing in low memory mode (`imemmode` = 1), extraction of the true derivatives is not possible, and POSTFLOW will automatically set `itruev` = 0 and echo a warning message to the screen in this case. Possible values for `itruev` are:

0	evaluate derivatives using a 1st–order approximation
1	evaluate derivatives using accurate 2nd–order expressions

`inrest`

Specify the format of the restart file. This can be any of the formats written by DPLR, summarized here:

1	parallel archival file (native unformatted)
11	parallel archival file (XDR format)
21	parallel archival file (ASCII)

See Appendix A for a complete list and description of the various file formats supported by the DPLR software package.

`ingrid`

Specify the format of the grid file. This can be any of the formats usable by DPLR, with values as given above for `inrest`. In addition (and preferably), the user can specify `ingrid` = 0, which tells POSTFLOW that the format and name of the grid file should be determined by reading the pertinent information from

the restart file. Using `ingrid` = 0 helps to ensure consistency when post-processing a dataset. The user is ensured that the grid file being used to post-process the data is the same as the one used to generate the data in the first place. However, if the name of the grid file, or its location relative to the restart file, is ever changed, this method will not work. In this case, the user can still specify the correct grid file by setting the appropriate value for `ingrid`. At this time, POSTFLOW does not perform any internal consistency checks (such as computation of a checksum) to ensure that the grid file used to post-process data is identical to that used to generate the data in the first place. Possible settings of `ingrid` are:

0	get format from restart file
1	parallel archival file (native unformatted)
11	parallel archival file (XDR format)
21	parallel archival file (ASCII)

`inbcf`

Specify the format of the boundary condition (BC) file, if any. This can be any of the formats usable by DPLR, or can be set to zero, indicating (as with `ingrid` above), that the format and name of any required BC file are to be determined from the restart file. POSTFLOW will determine whether a BC file is required by polling the restart file. If no BC file was used during the simulation one is not required for post-processing, and the value of `inbcf` is ignored. Possible settings of `inbcf` are:

0	get format from restart file
1	parallel archival file (native unformatted)
11	parallel archival file (XDR format)
21	parallel archival file (ASCII)

`ouform`

Specify the desired format of the output data. Possible settings of `ouform` are:

2	plot3d grid or q-file (native unformatted)
3	plot3d grid or function file (native unformatted)
5	Tecplot block binary
6	Tecplot point binary
7	compute max/min values for variables and output to STDOUT
8	sum variables over given surface(s) and output to STDOUT
9	RESERVED
10	print selected freestream quantities to STDOUT

11	output datasets for ***Moment*** calculations
17	compute max/min & maxloc/minloc and output to STDOUT
18	print a list of NaN locations to STDOUT
22	plot3d grid or q-file (ASCII)
23	plot3d grid or function file (ASCII)
25	Tecplot block ASCII
26	Tecplot point ASCII
32	gzipped plot3d grid or q-file (ASCII)
33	gzipped plot3d grid or function file (ASCII)
110	print freestream quantities to STDOUT in tabular format

Several of these options require more discussion, which is provided in subsequent sections. The primary output formats for datasets to be written for further post-processing (with Tecplot®, FAST, or another post-processing tool), are the plot3d and Tecplot formats listed above (`ouform` = 2:6, 22:26, and 32:33).

`iwrtd`

One of the more powerful features of POSTFLOW is the ability to recreate usable CFD input and physical data decks directly from the restart file. This is because, in addition to the pointwise flow data stored in this file, a section of the file is used to store all of the flags read from the CFD input deck and all of the physical modeling coefficients used to perform the simulation. Because of this, it is always possible to determine the settings and physical constants used to generate the simulation, even if the CFD input deck has been altered or misplaced. Setting `iwrtd` = 1 will cause POSTFLOW to create a subdirectory named "INPUTDECKS" in the current working directory. After the POSTFLOW run is complete this directory will contain a reconstruction of the CFD input deck that was used to run the simulation, as well as all of the physical property data decks read during execution. These files contain only the data actually used during the simulation (for example, physical properties for CO_2 will not appear in this reconstructed deck unless CO_2 was one of the species being modeled in the simulation). However, these files are formatted correctly, and can be directly used to conduct further simulations if desired. Input deck reconstruction can be performed in concert with any of the other options in POSTFLOW. Although POSTFLOW is capable of processing restart files generated by previous versions of DPLR, the CFD input deck will always be generated in the current format. If desired, the utility ***dpconvert*** can be used to change the format of the input deck after it is created. See Appendix U for more information on the use of ***dpconvert***. Possible settings of `iwrtd` are:

0	do not reconstruct input decks
1	reconstruct input decks

`interp`

This flag controls the way cell-centered finite-volume flow data is represented on a node-centered grid. Three methods are provided in POSTFLOW:

0	move flow data to the lower-left cell
1	interpolate grid points to cell centers
2	interpolate flow data to grid points
11	interpolate grid points to cell centers; no boundary points
21	interpolate grid points to cell centers; even at boundaries

The simplest choice is to simply move the cell-centered data to the "lower-left" grid point (`interp` = 0). No interpolation of the flow data or the grid is performed. This is not generally a desirable option, since the output data are slightly distorted and boundary data are presented incorrectly, but is provided for compatibility with heritage codes.

The second option is to generate a cell-centered grid and add additional face-centered points along the boundaries (`interp` = 1). In this option the flow quantities are not interpolated in the interior of the grid, and thus distortion of the output data is held to a minimum. However, flow properties are interpolated to the face centers along the boundaries in order to correctly reproduce face-centered boundary conditions. In this option the output data size defaults to the number of cells in a computational direction, plus two points in each direction representing the points added along the boundaries. When using this option it is important to remember that the output grid points lie at the cell centers of the original CFD grid, and thus the output grid cannot be used to run further CFD simulations.

The third interpolation option preserves the location of the CFD grid points and interpolates the finite-volume data onto these mesh points (`interp` = 2). This option is best when it is important that the locations of the CFD grid points be preserved in the output dataset. One example when this is important is when the output data is to be processed in ***SAGe*** or ***OutBound*** in order to move the outer boundary of the grid or adapt the grid to the computed flowfield.

The fourth interpolation option (`interp` = 11) is identical to `interp` = 1 discussed above, except that additional points are not added at the block boundaries. The maximum output dimensions using this option is the number of cells in each computational direction. When using this option it is important to note that the output grid will have “holes” in it along block boundaries, since no output is generated at the boundary itself. This output format is primarily used for computing integrated forces and moments, or for outputting pointwise forces for later offline integration.

The fifth interpolation option (`interp` = 21) is also identical to `interp` = 1 discussed above, except that in this case even the points at the boundaries are located using cell-centered interpolation. The maximum output dimensions using this option is the number of cells in each computational direction, plus two points in each direction representing the points added within the boundaries. This option is seldom used in practice; its primary purpose is for code developers to gain access to the cell centered values of quantities in the grid dummy cells (rather than the face-centered values available using `interp` = 1) for debugging purposes.

`nzones`

This is an integer that specifies the maximum number of output data zones to be generated. This is primarily a holdover from pre-F90 days and is used by the code to size certain output arrays. For most problems setting `nzones` equal to a moderate number such as 20 should suffice. If the value is too small, the program will abort and an error message will be generated prompting the user to increase the value.

`isep`

This flag controls whether multiple output datasets are to be written to a single or multiple files. Possible settings of `isep` are:

0	all active output datasets are written to a single file
1	each active output dataset is written to its own file

`istyp`

This flag is provided for future expansion and currently is not used in POSTFLOW.

`lref`

This is the reference length, used only for the normalization of moment coefficients. The extraction of moments and moment coefficients can either be performed directly in POSTFLOW or via an included utility program ***Moment***. The value of `lref` is passed to ***Moment*** in this case.

`aref`

This is the reference area, used only for the normalization of force and moment coefficients.

xmc,ymc,zmc

These define the *xyz* position of the moment reference center, used for extracting moments and moment coefficients.

imrx,imry,imrz

These flags are used to define symmetries in the simulation. This is useful in the computation of integrated forces and moments. For example, if a simulation is performed of a bilaterally symmetric vehicle at angle of attack, but with zero sideslip, it is typical to simulate 1/2 of the vehicle. By taking advantage of this bilateral symmetry the solution can be obtained with half of the grid points required for the full simulation. By definition such a vehicle will have zero net side force, since the total side force on the portion of the vehicle that is simulated will be exactly balanced by an equal and opposite force on the other half. However, when extracting force and moment coefficients from the simulation, it is necessary to provide this information to POSTFLOW in order for the resultant forces to be computed correctly. This is done through the use of the `imrx`, `imry`, and `imrz` flags, which define plane(s) of symmetry in the simulation. Currently supported are bilateral (any one of `imrx`, `imry`, or `imrz` are non-zero), and quadrilateral (two non-zero components) symmetry, where the axes of symmetry are aligned with the coordinate axes. If the symmetry of the vehicle is more complex than this, the user must set all of `imrx`, `imry`, and `imrz` to zero and compute the symmetry relations off line after post-processing is complete. Each of the symmetry flags can either be set to 1 (enforce symmetry about this plane), or 0 (do not enforce symmetry). The possible planes of symmetry are defined as:

`imrx` → body is symmetric about the *yz*-plane
`imry` → body is symmetric about the *xz*-plane
`imrz` → body is symmetric about the *xy*-plane

As an example of the function of these symmetry flags, consider a vehicle oriented in standard aircraft coordinates that is symmetric about the *xz*-plane. By setting `imry` = 1, POSTFLOW will first compute any requested forces. If integrated forces or moments are requested as output (`ouform` = 8), POSTFLOW will perform the integration, double the computed value of the forces in the *x* and *z* directions, and zero the value in the *y*-direction. The final integrated force or moment output will then be accurate for the entire vehicle, even though only half of the vehicle was simulated. Note that if force or moment coefficients are

requested as output it is important to use the full reference area when normalizing these computed forces if the symmetry flags are used.

All three of the symmetry flags are valid for 3D flows, and none are valid for axisymmetric flows. Since a 2D flow is assumed to lie in the *xy*-plane in DPLR, `imrz` has no meaning for a 2D flow simulation.

`iwind`

This flag is used to define a global "wind" axis for output. Possible values of `iwind` are:

0	do not alter the raw output data
1	determine sign by a dot product with freestream vector
2	determine sign by a dot product with supplied wind vector

The global wind axis is used either to determine the sign of the output skin friction (shear stress) or to convert output forces into a wind-based (lift and drag) coordinate system. Unless one of these outputs is requested as output the input value of `iwind` will be ignored. The orientation of the wind vector for `iwind` = 2 is defined using the `cxs`, `cys`, and `czs` flags defined below.

`cxs,cys,czs`

Defines the "cosines" of the global wind axis in the *xyz* directions when `iwind` = 2. These are defined as unit metrics, such that

$$\texttt{cxs}^2 + \texttt{cys}^2 + \texttt{czs}^2 = 1$$

Input values are always normalized to ensure that this expression is valid.

`iexbc`

Integer array that defines one or more surface zones to extract from the dataset. `iexbc` is a comma or space separated list of valid boundary condition (BC) numbers. See Section XX of the DPLR Users Manual for a listing of the valid BC numbers in DPLR. When POSTFLOW is run, any valid BC numbers specified by the `iexbc` flag will automatically be extracted from the dataset. If multiple instances of the BC number(s) exist, the resulting data will be saved as separate blocks (for plot3d output) or zones (for Tecplot® output). These zones will either be concatenated together into a single file or stored as separate files, depending on the setting of the `isep` flag. If Tecplot output is specified, output zones will be

named according to the BC number extracted, eg "BC14". Surface extraction can be used in conjunction with or instead of zone specification extraction, defined below. Surface extraction is a quick, foolproof, and easy method to extract defined surfaces from a complex multiblock grid, and should be used whenever possible to simplify the extraction process. Note that at the current time surface extraction cannot be used to extract surfaces that are defined with pointwise boundary conditions in an input BC file, although you can choose to extract all pointwise boundaries by setting `iexbc` = 0. If surface extraction is not desired, `iexbc` should be set to –1, which disables this feature. See below for more information on ways of extracting data using POSTFLOW.

`ivarp`

Integer array that defines the flow variables to be extracted from the restart file. This array is expressed as a comma or space separated string of integers representing the desired flow variables. Where possible, standard plot3d (or GASP®) variable numbers are used to represent flow quantities, although POSTFLOW allows extraction of a considerable superset of the variables available in PLOT3D or GASP.

Not all variables can be extracted in all circumstances. For instance, the *w*-component of the velocity vector cannot be extracted from a 2D or axisymmetric flowfield and the coefficient of viscosity cannot be extracted from an Euler simulation. If a variable is selected for extraction that is not permitted, it will be removed from the variable list automatically by POSTFLOW, and an informational message will be echoed to the screen. Each output variable also has a unique character representation, indicated below in parenthesis. This representation is used to name the variables when Tecplot or freestream output datasets are specified. Note that character representations are case-independent for compatibility with Tecplot.

POSTFLOW also permits a "shorthand" notation that allows the extraction of several related variables with a single number. For example, selecting `ivarp` = 1000 instructs POSTFLOW to output species densities for all species in the simulation. In all cases when shorthand notation is used only those variables relevant to the current case will be extracted. For example, the shorthand `ivarp` = 0 will automatically extract *x*, *y*, and *z* for a 3D flow, but only *x* and *y* for a 2D or axisymmetric flow. All of these "shorthand" selections are indicated below.

All extracted variables are output in SI units. Variable numbers listed as "RESERVED" below are not currently implemented in POSTFLOW, but have been allocated for future expansion. Entries prefaced with an asterisk (*) in the list below are defined as surface-specific quantities. These quantities are extracted with respect to a given surface direction, defined either with the `ifac` flag in the

zone specifications (see below), or automatically determined when extracting surfaces with `iexbc`. Possible values for `ivarp` are given below in list form grouped by category; see Appendix P for a more detailed description of many of the variables.

Grid Coordinates

0 all grid coordinates
1 *x*-coordinate (x)
2 *y*-coordinate (y)
3 *z*-coordinate (z)

Grid-Related Variables

10 all path-lengths
11 path length along grid lines in *i*-direction (si)
12 path length along grid lines in *j*-direction (sj)
13 path length along grid lines in *k*-direction (sk)
14 *unit outward normal *x*-direction cosine (sx)
15 *unit outward normal *y*-direction cosine (sy)
16 *unit outward normal *z*-direction cosine (sz)

21 *body normal distance (dn)
22 *deviation from orthogonality [deg.] (dev)
23 *face area (Area)
25 maximum cell aspect ratio (CAR)

Mixture Transport Properties

50 total viscosity (mu)
51 total kinematic viscosity (nu)
52 total translational thermal conductivity (kap)
53 total rotational thermal conductivity (kapr)
54 total vibrational thermal conductivity (kapv)
55 free electron thermal conductivity (kape)
56 total binary diffusion coefficient (D)
57 mixture mean free path (mfp)
58 unit Reynolds number (Re/L)
59 cell Reynolds number (Re_c)

Thermodynamic Properties

60 ratio of frozen specific heats cp/cv (G)

61 frozen specific heat at constant volume (cv)
62 frozen specific heat at constant pressure (cp)
63 translational specific heat at constant volume (cvt)
64 rotational specific heat at constant volume (cvr)
65 vibrational specific heat at constant volume (cvv)
66 electronic specific heat at constant volume (cve)
68 mixture gas constant (R)
69 mixture molecular weight (Mw)

Turbulence Quantities

70 turbulent kinetic energy (TKE)
71 turbulent omega (omega_t)
72 RESERVED
73 RESERVED
75 Spalart-Almaras conserved variable (mu_SA)

Laminar Transport Properties

80 laminar viscosity (mu_l)
81 laminar kinematic viscosity (nu_l)
82 laminar thermal conductivity (kap_l)
83 laminar rotational thermal conductivity (kapr_l)
84 laminar vibrational thermal conductivity (kapv_l)
85 laminar free electron thermal conductivity (kape_l)
86 laminar binary diffusion coefficient (D_l)
87 laminar Lewis number (Le)
88 laminar Schmidt number (Sc)
89 laminar Prandtl number (Pr)

Turbulent Transport Properties

90 turbulent eddy viscosity (mu_t)
91 turbulent kinematic eddy viscosity (nu_t)
92 turbulent thermal conductivity (kap_t)
93 turbulent rotational thermal conductivity (kapr_t)
94 turbulent vibrational thermal conductivity (kapv_t)
95 turbulent free electron thermal conductivity (kape_t)
96 turbulent binary diffusion coefficient (D_t)
97 turbulent Lewis number (Le_t)
98 turbulent Schmidt number (Sc_t)
99 turbulent Prandtl number (Pr_t)

Mixture Flow Properties

Stagnation quantities (density, pressure, and temperature) are computed assuming isentropic relations, and thus are not valid for a chemically reacting flowfield.

100 mixture density (rho)
101 mixture number density (N_tot)
102 stagnation mixture density (r_o)

110 pressure (p)
111 dynamic pressure (Q)
112 stagnation pressure (p_o)
113 Pitot pressure (p_pitot)
114 pressure coefficient (C_p)

120 translational temperature (T)
121 bulk temperature (T_b)
122 stagnation temperature (T_o)
124 rotational temperature (Tr)
125 vibrational temperature (Tv)
126 electronic temperature (Te)
127 free electron temperature (Tel)

132 total enthalpy per unit mass (h)
133 static enthalpy per unit mass (h_s)
134 total enthalpy per unit volume (rh)
135 static enthalpy per unit volume (rh_s)

142 total energy per unit mass (e)
143 total translational energy per unit mass (et)
144 total rotational energy per unit mass (er)
145 total vibrational energy per unit mass (ev)
146 total electronic energy per unit mass (ee)
147 total free electron energy per unit mass (eel)
148 total chemical formation energy per unit mass (eh)
149 total kinetic energy per unit mass (eU)

150 velocity in the *x*-direction (u)
151 velocity in the *y*-direction (v)
152 velocity in the *z*-direction (w)
153 velocity magnitude (Vel)
154 frozen Mach number (M)
155 frozen sound speed (a)
156 mean thermal speed (cbar)
157 normalized velocity in the *x*-direction (u/Vel)
158 normalized velocity in the *y*-direction (v/Vel)

159 normalized velocity in the *z*-direction (w/Vel)

160 momentum per unit volume in the *x*-direction (rhou)
161 momentum per unit volume in the *y*-direction (rhov)
162 momentum per unit volume in the *z*-direction (rhow)
163 total energy per unit volume (re)
164 total rotational energy per unit volume (rer)
165 total vibrational energy per unit volume (rev)
166 total electronic energy per unit volume (ree)
167 total free electron energy per unit volume (rel)
168 total chemical formation energy per unit volume (reh)
169 total kinetic energy per unit volume (reU)

170 entropy (S)
175 pointwise unit radiative emission (Erad)

180 degree of ionization (zeta)
181 debye length (lam_D)
182 Tstar (Tstar)

194 total energy per unit mass in rotational Eqn. (er_B)
195 total energy per unit mass in vibrational Eqn. (ev_B)
196 total energy per unit mass in electronic Eqn. (ee_B)
197 total energy per unit mass in free electron Eqn. (el_B)

202 *delta velocity at wall (Del_V)
204 *delta temperature at wall (Del_T)

250 velocity in the x-direction normalized by V_∞ (u/Vin)
251 velocity in the y-direction normalized by V_∞ (v/Vin)
252 velocity in the z-direction normalized by V_∞ (w/Vin)

324 limited rotational temperature (Tr_l)
325 limited vibrational temperature (Tv_l)
326 limited electronic temperature (Te_l)
327 limited free electron temperature (Tel_l)

Viscous Derivative-Based Quantities

501 *skin friction coefficient (Cf)
507 *total wall shear stress (tau)

511 *Stanton number [based on wall enthalpy] (Ch)
512 *Heat transfer coefficient in mass flux units (Chm)
517 *Stanton number [based on freestream conditions] (St)

518 *Convective heating coefficient (Ct)

520 radiative equilibrium heat transfer (Qeq)
521 *total wall heat transfer (qw)
522 *translational wall heat transfer (qT)
523 *rotational wall heat transfer (qR)
524 *vibrational wall heat transfer (qV)
525 *free electron wall heat transfer (qEl)
526 *catalytic wall heat transfer (qD)
527 *velocity wall heat transfer (qU)

581 *spacing in wall units y^+ (yp)
584 *inner velocity u^+ (up)

591 *blowing velocity through face (vb)
594 *mass flow rate through face (mdot)
595 *unit mass flow rate through face (mdotU)

Aerodynamic Forces and Moments

600 *total force on a face in all directions
601 *total force on a face in *x*-direction (Fx)
602 *total force on a face in *y*-direction (Fy)
603 *total force on a face in *z*-direction (Fz)
604 *total force on a face in *x*-direction per unit area (Fx_a)
605 *total force on a face in *y*-direction per unit area (Fy_a)
606 *total force on a face in *z*-direction per unit area (Fz_a)

610 *pressure force on a face in all directions
611 *pressure force on a face in *x*-direction (Fx_P)
612 *pressure force on a face in *y*-direction (Fy_P)
613 *pressure force on a face in *z*-direction (Fz_P)
614 *pressure force on a face in *x*-direction per unit area (Fx_Pa)
615 *pressure force on a face in *y*-direction per unit area (Fy_Pa)
616 *pressure force on a face in *z*-direction per unit area (Fz_Pa)

620 *viscous force on a face in all directions
621 *viscous force on a face in *x*-direction (Fx_V)
622 *viscous force on a face in *y*-direction (Fy_V)
623 *viscous force on a face in *z*-direction (Fz_V)
624 *viscous force on a face in *x*-direction per unit area (Fx_Va)
625 *viscous force on a face in *y*-direction per unit area (Fy_Va)
626 *viscous force on a face in *z*-direction per unit area (Fz_Va)

650 *total force coefficient on a face in all directions

651 *total force coefficient on a face in *x*-direction (Cx)
652 *total force coefficient on a face in *y*-direction (Cy)
653 *total force coefficient on a face in *z*-direction (Cz)

660 *pressure force coefficient on a face in all direction
661 *pressure force coefficient on a face in *x*-direction (Cx_P)
662 *pressure force coefficient on a face in *y*-direction (Cy_P)
663 *pressure force coefficient on a face in *z*-direction (Cz_P)

670 *viscous force coefficient on a face in all direction
671 *viscous force coefficient on a face in *x*-direction (Cx_V)
672 *viscous force coefficient on a face in *y*-direction (Cy_V)
673 *viscous force coefficient on a face in *z*-direction (Cz_V)

700 *total moment on a face in all directions
701 *total moment on a face in *x*-direction (Mx)
702 *total moment on a face in *y*-direction (My)
703 *total moment on a face in *z*-direction (Mz)

710 *pressure moment on a face in all directions
711 *pressure moment on a face in *x*-direction (Mx_P)
712 *pressure moment on a face in *y*-direction (My_P)
713 *pressure moment on a face in *z*-direction (Mz_P)

720 *viscous moment on a face in all directions
721 *viscous moment on a face in *x*-direction (Mx_V)
722 *viscous moment on a face in *y*-direction (My_V)
723 *viscous moment on a face in *z*-direction (Mz_V)

750 *total moment coefficient on a face in all directions
751 *total moment coefficient on a face in *x*-direction (Cmx)
752 *total moment coefficient on a face in *y*-direction (Cmy)
753 *total moment coefficient on a face in *z*-direction (Cmz)

760 *pressure moment coefficient on a face in all direction
761 *pressure moment coefficient on a face in *x*-direction (Cmx_P)
762 *pressure moment coefficient on a face in *y*-direction (Cmy_P)
763 *pressure moment coefficient on a face in *z*-direction (Cmz_P)

770 *viscous moment coefficient on a face in all direction
771 *viscous moment coefficient on a face in *x*-direction (Cmx_V)
772 *viscous moment coefficient on a face in *y*-direction (Cmy_V)
773 *viscous moment coefficient on a face in *z*-direction (Cmz_V)

Debugging/Status Information

990 pointwise BC numbers along block edges (ibcp)
991 net charge [should always be zero] (Qnet)
992 sum of mass fractions [should always be one] (Csum)
998 zero (zero)
999 pointwise residual (res)

Species Data

The following variables are species-specific data. In each case the user can choose to extract data for either a subset of the species by entering just the desired variable numbers, or data for all species by entering the appropriate "shorthand" number.

1000 all species densities
1000+*n* density of species *n* (n)

1200 all species number densities
1200+*n* number density of species *n* (N_n)

1400 all species mass fractions
1400+*n* mass fraction of species *n* (C_n)

1600 all species mole fractions
1600+*n* mole fraction of species *n* (X_n)

1800 all species densities, normalized by ρ_∞
1800+*n* normalized density of species n (RnD_n)

3400 all species rotational temperatures
3400+*n* rotational temperature of species *n* (Tr_n)

3600 all species vibrational temperatures
3600+*n* vibrational temperature of species *n* (Tv_n)

4000 all species total internal energies per unit mass
4000+*n* total internal energy per unit mass of species *n* (e_n)

4200 all species translational internal energies per unit mass
4200+*n* trans. internal energy per unit mass of species *n* (et_n)

4400 all species rotational internal energies per unit mass
4400+*n* rotational internal energy per unit mass of species *n* (er_n)

4600 all species vibrational energies per unit mass

4600+*n*	vibrational energy per unit mass of species *n* (ev_n)
4800	all species electronic energies per unit mass
4800+*n*	electronic internal energy per unit mass of species *n* (ee_n)
5000	*all species mass flow rates through surface
5000+*n*	*mass flow rate through surface of species *n* (mdot_n)
5200	*all species mass flow rates through surface [per unit area]
5200+*n*	*mass flow rate through surface of species *n* (mdotU_n)
6000	all species total specific heats at constant volume
6000+*n*	total specific heat at constant volume of species *n* (cvx_n)
6200	all species translational specific heats at constant volume
6200+*n*	translational specific heat at const. vol. of species *n* (cvt_n)
6400	all species rotational specific heats at constant volume
6400+*n*	rotational specific heat at const. vol. of species *n* (cvr_n)
6600	all species vibrational specific heats at constant volume
6600+*n*	vibrational specific heat at const. vol. of species *n* (cvv_n)
6800	all species electronic specific heats at constant volume
6800+*n*	electronic specific heat at const. vol. of species *n* (cve_n)
7000	all species frozen specific heats at constant pressure
7000+*n*	specific heat at constant pressure of species *n* (cp_n)
7200	all species frozen specific heats at constant volume
7200+*n*	specific heat at constant volume of species *n* (cv_n)
8000	all species gas constants
8000+*n*	gas constant of species *n* (R_n)
8200	all species equivalent degrees of freedom [nkT]
8200+*n*	equivalent degrees of freedom of species *n* (dof_n)
8400	all species partial pressures
8400+*n*	partial pressure of species *n* (p_n)
8600	all species mean thermal speeds
8600+*n*	mean thermal speed of species *n* (cbar_n)
8800	all species chemical formation energies per unit mass

8800+*n*	formation energy per unit mass of species *n* (eh_n)
10000	all species diffusion coefficients
10000+*n*	diffusion coefficient of species *n* (D_n)
10200	all species ambipolar diffusion effectiveness
10200+*n*	ambipolar diffusion effectiveness of species *n* (DaC_n)
10400	all species effective Schmidt numbers
10400+*n*	effective Schmidt number of species *n* (Sc_n)
10800	all species unit diffusion mass fluxes
10800+*n*	unit diffusion mass flux of species *n* (MD_n)

Tecplot/Plot3D Zone Specification

Following the `ivarp` flag are a series of flags that are used to define the extents of desired data extractions. In general one row of data is used to define each desired extraction. The last line in this group is the so-called "terminator" line, in which the first entry (`iwrt`) is set equal to -1. This terminator line instructs the code to stop reading zone specification information, and thus it must be present or a runtime error will occur. The flags in the zone output specification lines are defined as follows:

`iwrt`

This flag determines whether the zone specification on that line will actually be extracted by POSTFLOW. Possible settings of `iwrt` are:

0	do not extract the data defined by this zone specification
1	extract the data defined by this zone specification
-1	terminator line

The user is permitted to enter any number of zone specification lines in the input deck. However, only those that are "turned on" (`iwrt` = 1) will actually be extracted at runtime. In this way the user can set up a default input deck with multiple zone specification lines for all possible desired output. Then each time the post-processor is run only the data that are actually required are "turned on", the rest are left inactive.

`ifac`

Defines the *ijk* orientation of the surface being extracted. This is used only when surface-oriented quantities (those marked with an asterisk in the definition of `ivarp` above, such as skin friction or heat transfer) are desired for output, and allows the code to determine which surface to output data to. Possible values of `ifac` are:

0	No face selected. Surface-oriented output will not be printed
1	*i*-face
2	*j*-face
3	*k*-face

`imin,imax,jmin,jmax,kmin,kmax`

Integer values that define the extent of the desired extraction in the *ijk* directions. Numbering is dependent on the value of `interp`, but in each case begins with 1. For example, in the case of a single block 2D grid of size 65×129 grid points (64×128 cells), the full extent of *ijk* would be as follows:

`interp` = 1 → `imin` = 1
`imax` = 66
`jmin` = 1
`jmax` = 130

The maximum values in each case are the number of cells in that direction + 2, since points will be added to the output dataset at the face centers of each grid boundary when `interp` = 1.

`interp` = 2 → `imin` = 1
`imax` = 65
`jmin` = 1
`jmax` = 129

The maximum values in each case are the number of points, since for `interp` = 2 the flow quantities have been interpolated to the existing grid points.

It is not necessary for the user to determine the actual size of each block before extraction. POSTFLOW recognizes the ``shorthand" value of -1 to indicate the maximum possible value in any of the coordinate directions. For example, specifying `imin` = 1 and `imax` = -1 instructs POSTFLOW to extract all *i*-values from the given dataset, regardless of the value of `interp`. Other shorthand values that can be used are -2 (which is equivalent to the maximum value -1), and -3 (which is equivalent to the mid-point). Finally, if a plane of data is desired, simply set the minimum and maximum values in that direction to be the same. Examples of using the zone specification flags to output datasets will be given

below. Note that it is not currently possible to extract data with a stride other than 1; ie it is not possible to directly extract every OTHER *i* and *j* point from a dataset.

`bkmin,bkmax`

These integer quantities define the minimum and maximum grid block numbers from which to extract data. As before, a value of -1 indicates the maximum block number (ie. the last block in the simulation). In this way if the same extraction is to be performed over multiple blocks it can be defined just once.

`zonetitle`

This is an ASCII string surrounded by single or double quotes that will be used as the zone title if Tecplot output is specified. This can be set to a descriptive value in order to better define multiple output zones, but it is not necessary. If a zone title is not desired, zonetitle should simply be set to the empty string (“”).

I/O Filenames

The final section of the POSTFLOW input deck defines input/output filenames. All filenames must be enclosed in single or double quotes, and may include a relative or absolute pathname if desired. The first two (`fname` and `pname`) are required by POSTFLOW, but the final two filenames `gname` and `bname`) are only read if required.

`fname`

The name of the restart file to process.

`pname`

The name of the output file to create (if any). Some output formats echo data to the screen (STDOUT). In these cases the output filename is not used.

`gname`

The name of the grid file to process (used only if `ingrid` ≠ 0).

bname

The name of the BC file to process (used only if `inbcf` ≠ 0).

Extracting Volume or Surface Data

The primary use of POSTFLOW is to extract volume or surface data from the restart file for further post-processing or visualization. This data can be saved in either plot3d (`ouform` = 2, 3, 22, 23, 32, 33) or Tecplot (`ouform` = 5, 6, 25, 26) formats. Plot3d format is a standard CFD output format that can be read by most commercial post-processing tools, while Tecplot format is (of course) useful only if further post-processing is to be performed using Tecplot. POSTFLOW can write Tecplot ASCII (.dat) files as well as binary (.plt) files, although Tecplot binary output requires linking to the Amtec-provided "`tecio.a`" (or "`tecio64.a`") runtime library. If this library is not available on your machine, Tecplot binary files cannot be generated. Gzipped plot3d output (`ouform` = 32, 33) is generated via a system call to the **gzip** utility provided with UNIX and LINUX systems. This option may not be available on Windows systems.

Volume or surface data can be extracted using zone specification lines defined above, surface extraction with the `iexbc` flag, or a combination of the two. Variables desired for output are specified using the `ivarp` integer array, as discussed above. This use of POSTFLOW will be demonstrated here with a pair of examples.

For the first example, assume that a simulation was performed on a five-block 3D volume grid, and the desired output variables are pressure (`ivarp` = 110), temperature (`ivarp` = 120), Mach number (`ivarp` = 154), and pointwise residual (`ivarp` = 999). The `ivarp` array for this case is given as:

```
ivarp
110 120 154 999
```

If we wish to extract the entire volume of data, this can be accomplished with a single zone specification line:

```
iwrt ifac imin imax jmin jmax kmin kmax  bkmin  bkmax  zonetitle
 1,   0,    1,  -1,   1,  -1,   1,  -1,    1,    -1    'volume'
-1,   0,    1,  -1,   1,  -1,   1,  -1,    1,    -1 'terminator'
```

where we have simply specified, with the help of the -1 shorthand described previously, that we wish to extract all *ijk* points from all blocks. The value of `ifac` is set to zero, indicating that we are not extracting surface-oriented data. For this example POSTFLOW will generate five output zones, which will contain the entire volume. Each zone will be called "volume" if a Tecplot output file format is selected. The second line is the required terminator, which instructs POSTFLOW to stop reading zone specification lines.

Continuing with this example, let us further assume that all blocks have a body surface at $j = 1$, and that these five surfaces completely define the body. In this case we can extract the entire body surface with the following line:

```
iwrt ifac imin imax jmin jmax kmin kmax  bkmin  bkmax  zonetitle
 1,   2,    1,  -1,   1,   1,   1,  -1,    1,    -1    'body'
-1,   0,    1,  -1,   1,  -1,   1,  -1,    1,    -1 'terminator'
```

where we have extracted the $j = 1$ surface from all blocks (`jmin` = `jmax` = 1), and labeled the resulting zones "body". In this case we have also set `ifac` = 2, indicating that a j-surface is being extracted. As stated previously, the value of `ifac` is important only when derivative-based quantities are selected for output.

Now, assume further that the exit (outflow) plane of the problem can be completely defined as the `imax` surface of block #5. Extracting this surface is accomplished by the following specification:

```
iwrt ifac imin imax jmin jmax kmin kmax  bkmin  bkmax  zonetitle
 1,   1,   -1,  -1,   1,  -1,   1,  -1,    5,     5    'outflow'
-1,   0,    1,  -1,   1,  -1,   1,  -1,    1,    -1 'terminator'
```

If desired, all of these zone specification lines can be combined in a single input deck, and they can be selectively activated or inactivated each time POSTFLOW is run using the `iwrt` flag. For example, the following lines:

```
iwrt ifac imin imax jmin jmax kmin kmax  bkmin  bkmax  zonetitle
 0,   0,    1,  -1,   1,  -1,   1,  -1,    1,    -1   'volume'
 1,   2,    1,  -1,   1,   1,   1,  -1,    1,    -1   'body'
 1,   1,   -1,  -1,   1,  -1,   1,  -1,    5,     5   'outflow'
-1,   0,    1,  -1,   1,  -1,   1,  -1,    1,    -1 'terminator'
```

define all three output datasets discussed for this example, but since `iwrt` = 0 for the first specification only the body and outflow datasets will be generated when POSTFLOW is run.

For the second example we consider a four block grid, in which the body surface is defined by the `imin` plane of block #1, the `kmax` plane of block #3, and the `imin` and `jmax` planes of block #4. In this case, if we wish to extract the entire body surface using zone specification lines we must enter:

```
iwrt ifac imin imax jmin jmax kmin kmax  bkmin  bkmax  zonetitle
 1,   1,    1,   1,   1,  -1,   1,  -1,    1,     1    'surface'
 1,   3,    1,  -1,   1,  -1,  -1,  -1,    3,     3    'surface'
 1,   1,    1,   1,   1,  -1,   1,  -1,    4,     4    'surface'
 1,   2,    1,  -1,  -1,  -1,   1,  -1,    4,     4    'surface'
-1,   0,    1,  -1,   1,  -1,   1,  -1,    1,    -1 'terminator'
```

This approach will work, but it is cumbersome to set up, and requires the user to pre-determine the locations of all surface sub-zones in the simulation. An alternative is to use the `iexbc` flag to extract the body surface in a single step. Assuming that the body surface is catalytic and in radiative equilibrium (BC = 26), simply setting `iexbc` = 26 will automatically extract all four zones from the restart file.

Remember that the `iexbc` flag can be used together with the zone specification flags in a single POSTFLOW run. By using a combination of these methods it should be possible to extract almost any desired subset of the flowfield.

Finally, it is possible to use POSTFLOW to view the coordinates of the grid dummy cells if desired. This is accomplished by setting `interp` = 0 and `ivarp` = 0. This is provided mainly for debugging purposes, since generally the dummy cell information is meant to be transparent to the end user.

Extracting Zone Minima or Maxima

POSTFLOW can also be used to extract the minimum or maximum values of selected output variables in each output dataset, and, if desired, the *ijk* location of these values. There are two separate output formats provided to accomplish this. The first, `ouform` = 7, will display a listing of the minimum and maximum values of the selected variables in each output zone to STDOUT. The second option, `ouform` = 17, displays a longer listing to STDOUT, which includes the *ijk* locations of these maximum and minimum values in the zone. Note that the *ijk* location is computed relative to the output zone. If absolute *ijk* values are required the entire volume should be selected as output.

In each case the user specifies the desired output data using the `ivarp` array, and the output is written to STDOUT. An output datafile is not generated when min/max data are requested.

An example of the output of POSTFLOW is shown here for `ouform` = 7:

```
block # 1: nx =   32; ny =   16; nz =   64
      zone t=BC19          i=  34 j=   1 k=  66

 Zone Maximum and Minimum Values:
        p       [max] =  5.0043E+04;   [min] =  3.5910E+01
        T       [max] =  1.5345E+04;   [min] =  1.2807E+02
        M       [max] =  3.2322E+01;   [min] =  0.0000E+00

 processing grid variable  1  2  3
 processing flow variable  1  2  3  4  5  6  7  8  9

 block # 2: nx =   48; ny =   64; nz =   64
      zone t=BC19          i=  50 j=   1 k=  66
```

```
 Zone Maximum and Minimum Values:
        p        [max] =  4.4431E+04;   [min] =  3.5910E+01
        T        [max] =  1.4203E+04;   [min] =  1.2807E+02
        M        [max] =  3.2322E+01;   [min] =  0.0000E+00
```

Extracting Integrated Surface Data

Another use of POSTFLOW is the integration of surface quantities. This is exercised by setting `ouform` = 8, and ensuring that all output datasets define surfaces (either with `iexbc` or the `ifac` flag). When this option is specified, POSTFLOW will compute the panel area of each surface face in each valid output zone, and multiply this panel area by the desired integrated output variable(s). This value is then summed over each surface zone, and the result is presented on STDOUT. Results are shown for each zone, and a sum over all zones is also computed. If any of the symmetry flags (`imrx`, `imry`, `imrz`) have been selected, the summed output values are adjusted as discussed above to account for the symmetries of the problem. This option works only with `interp` = 11. The most common use of this option is the computation of integrated aerodynamic forces or moments.

At this time the only variables that can be extracted as integrated surface quantities are the face area (`ivarp` = 23), aerodynamic forces (`ivarp` = 600:673), aerodynamic moments (`ivarp` = 700:773), heat transfer (`ivarp` = 520:527), and mass flow rates (`ivarp` = 594:595). Any other selected variables will be removed from the list if they are selected when `ouform` = 8.

If aerodynamic forces are selected and `iwind` is set to either 1 or 2, output forces will be rotated into the wind coordinate system based on either the internal (`iwind` = 1) or provided (`iwind` = 2) velocity cosines, and will be output as lift, drag, and side forces in addition to the *xyz* forces otherwise reported. Note that this option assumes that the employed grid is in standard aircraft coordinates.

An example of the output for `ouform` = 8 is shown here:

```
block # 1: nx =   32; ny =   16; nz =   64
  ==> extracted derivative data from the KMIN-surface
  ==> derivative data computed using full viscous fluxes
      zone t=BC26          i=  32 j=  16 k=   1

        Fx      =  9.872234694840E+02   (N)
        Fy      =  3.249055280159E+02   (N)
        Fz      = -3.865734146780E+02   (N)

 processing grid variable  1  2  3
 processing flow variable  1  2  3  4  5  6  7  8  9

 block # 2: nx =   48; ny =   64; nz =   64
```

```
 ==> extracted derivative data from the KMIN-surface
 ==> derivative data computed using full viscous fluxes
     zone t=BC26         i=  48 j=  64 k=   1

       Fx      =  2.919904481514E+03   (N)
       Fy      =  1.605495325835E+04   (N)
       Fz      = -9.258734449289E+03   (N)


 Integrated Surface Quantities
 Summary Over All Output Surfaces:
 XZ-Symmetry Enforced During Final Summation

       Fx      =  7.814255901995E+03   (N)
       Fy      =  0.000000000000E+00   (N)
       Fz      = -1.929061572793E+04   (N)
```

Extracting Freestream Data

POSTFLOW provides two methods of extracting freestream data from the restart file. The first option, `ouform` = 10, will output an informational listing to STDOUT, displaying the freestream quantities requested and SI units of each. The second option, `ouform` = 110, displays a tabular listing of freestream data, which is better suited for direct import to a spreadsheet application.

In each case the user specifies the desired output data using the `ivarp` array, and the output is written to STDOUT. An output datafile is not generated when freestream data are requested. Freestream data are tabulated and output for each grid block in the simulation, regardless of any surface extraction or zone specification flags that have been set. Separate freestream data are presented for each grid block, since DPLR allows multiple freestream specifications to be applied when a simulation is run. However, in most cases all blocks will have the same freestream information.

As an example, for the "Neptune" sample problem, if we wish to extract freestream pressure, temperature, Mach number, and unit Reynolds number from the dataset we would specify:

```
ivarp
110 120 154 58
```

in the input deck. A portion of the output of POSTFLOW for this case is presented here for `ouform` = 10:

```
block # 1: nx =   32; ny =   16; nz =   64

 Freestream Quantities:
```

```
     Block # 1

        p        =  3.591044259306E+01   (Pa)
        T        =  1.280700000000E+02   (K)
        M        =  3.232180261501E+01   ()
        Re/L     =  3.151858720834E+05   (1/m)


 block # 2: nx =   48; ny =   64; nz =   64

     Block # 2

        p        =  3.591044259306E+01   (Pa)
        T        =  1.280700000000E+02   (K)
        M        =  3.232180261501E+01   ()
        Re/L     =  3.151858720834E+05   (1/m)
```

Extracting Data for Processing with *Moment*

POSTFLOW can now directly compute moments or moment coefficients. However, for historical reasons a utility called ***Moment*** is provided as part of the DPLR package that can also do this computation quite easily. ***Moment*** requires as input forces per unit area at each cell center on the surface of the vehicle; either total forces (`ivarp` = 604:606), pressure forces (`ivarp` = 614:616) or viscous forces (`ivarp` = 624:626). This is exercised by setting `ouform` = 11, and ensuring that all output datasets define surfaces (either with `iexbc` or the `ifac` flag). If any of the symmetry flags (`imrx`, `imry`, `imrz`) have been selected, the summed output values are adjusted as discussed above to account for the symmetries of the problem. This option works only with `interp` = 11.

When this option is run, plot3d grid and function files will be created, along with a file "Moment.inp", which is the input deck for the ***Moment*** utility. ***Moment*** is then run simply from the command line by typing

```
Moment < Moment.inp
```

A sample of the output from the ***Moment*** script is given here:

```
running Moment version 3.05.0
 ----------------------------------------

 Moment Center:
   Xm =  0.000000E+00 (m)
   Ym =  0.000000E+00 (m)
   Zm =  0.000000E+00 (m)
```

```
Reference Values:
  lref =  3.650000E+00 (m)
  aref =  4.500000E+00 (m^2)
  qdyn =  2.784862E+03 (Pa)

Vehicle Symmetries:
  xy-plane

Wetted Area:
  Area =  0.000000E+00 (m^2)

Force components:
  Fx  =  1.777037E+07 (N)     ;     Cx  =  1.418013E+03
  Fy  = -1.165808E+04 (N)     ;     Cy  = -9.302740E-01
  Fz  =  0.000000E+00 (N)     ;     Cz  =  0.000000E+00

Moment components:
  Mx  =  0.000000E+00 (N*m)   ;     Cmx =  0.000000E+00
  My  =  0.000000E+00 (N*m)   ;     Cmy =  0.000000E+00
  Mz  = -6.500303E+04 (N*m)   ;     Cmz = -1.421099E+00
```

Note that at this time there is no error checking in place to ensure that this output format is used correctly. In other words, it is not an error to select other variables as output, but the results generated by the ***Moment*** utility will be incorrect unless forces per unit area are selected.

Extracting NaN's from the Dataset

The final use of POSTFLOW is to extract the locations of any NaN's in the restart file. This is selected using `ouform` = 18, and is provided solely for debugging purposes. The intended purpose is to stop the simulation after a NaN has been generated, write out the restart file, and then post-process the file to determine where the NaN occurred. Note that once a NaN is generated by DPLR, it will quickly be convected throughout the solution domain, so if it is desired to view the location where the NaN first occurred it is important to stop the simulation and write a restart file at the conclusion of the iteration in which the NaN was first generated (this is typically the iteration PRIOR to when the residual itself becomes NaN).

Note that different machine architectures (and FORTRAN compilers) treat NaN's differently. In many cases generation of a NaN is a fatal error that will cause the program to abort immediately. In this case it is not possible to have a NaN in a restart file, since DPLR would have aborted before the value was written. However, it is usually possible to alter this behavior by recompiling with the appropriate compiler flags, if desired. Consult the man pages or reference manual for your compiler for more information.

In practice this is a rarely used option, but it can be handy to locate the occasional evil bug. The output data consists of a list of *ijk* locations of all NaN's in the volume, listed block-by-block to STDOUT.

APPENDIX: Release Notes for Version 3.05.0

UPGRADES:

v3.05.0 -- add new output format 11 for offline Moment calculations
-- add extraction of surface BC intersections
-- add variables 67,127,147,167,197,327
-- move old variables 147,148,167,168
-- add moment extraction (variables 700:773) [Matt MacLean, CUBRC]
-- add forces in wind coordinate system

BUGLIST:

v3.05.0 -- FIXED BUG: output millikan.vib file unreadable (readdeck)
-- FIXED BUG: edge heat transfer bad in cases where two surfaces abut (interpol)
-- FIXED BUG: not writing correct catalysis.surf file for case where material map present (readdeck)
-- FIXED BUG: supersonic Pitot pressure formula was wrong (writeflow)

MODLIST:

v3.05.0 -- add new output format 11 for Moment calculations (readinp,writeflow)
-- mods for multiple temp range emissivity (readdeck,writeflow)
-- add uncoupled Spalart-Almaras (parseivar,writeflow,readdeck)
-- remove variable ivarp=76 (writeflow,parseivar,setupcvar)
-- better memory error trapping (readdeck)
-- add Chapman viscosity routine (readdeck,premake)
-- add extraction of surface BC intersections (premake,readinp)
-- minor mods for electronic energy stuff (readdeck, surface, rdfiles, writeflow, fullvis, getsize)
-- add variables 67,127,147,167,197,327 (setupcvar,writeflow)
-- move old variables 147,148,167,168 (setupcvar,writeflow)
-- eliminate getef298 routine (readdeck)
-- add moment extraction capability [Matt MacLean, CUBRC] (main, writeflow, setupcvar, parseivar)
-- add forces in wind coordinate system (writeflow)